Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 87 (2016) 210 – 214

Procedia Computer Science

4th International Conference on Recent Trends in Computer Science & Engineering

PLANT:Permission Leakage AvoidaNce Through Filteration

M.Kireet[a], Dr.Meda Sreenivasa Rao[b]

[a]*Lecturer,Dept of CSE, JNTUCEH, Hyderabad,India*

[b]*Professor, Dept of CSE, JNTUSIT, Hyderabad, India*

Abstract

The instantaneous growth of mobile apps in mobile markets elevating the concerns on the protection of user's sensitive data in Smartphone's. The Majority of apps in all mobile markets are gathering the data which is insignificant to the app functionality. The mobile app collects the data of the user either users' sensitive data or device data by using orthodox permission control policy.This Permission policy in apps takes acceptance of user for installing an app and user has only one option to accept all permissions to continue the installation of app. In this work we proposed a framework PLANT which is Permission Leakage AvoidaNce Through Filteration for identifying data leakage done by mobile apps by using permissions .Finally framework gives the knowledge to the user to deny if there are any irrelevant permissions required by the app based on its functionality

Keywords: *PLANT; Data Leakage; Filteration*

1. Introduction

The expansion of Smartphone's throughout the world has extended the use of mobile apps.Almost 13% of today's e-commerce transactions are done using mobiles[3] and 50% of the online orders are placed by mobile apps. The number of apps abruptly increasing in the app stores. At present 2.6 billion people are using Smartphone's and by 2020 the Smartphone's usage may rise to 6.1 billion[4].Till 2015 more than 100 billion apps were downloaded from different app stores[4].Due to this sudden rise in usage of apps in Smartphone's several concerns raised on the privacy of user data in Smartphone's. By analysis [3] 98% of the mobile apps are vulnerable to security attacks which are the apps of top 50 e-commerce companies in India. Most of the apps follow orthodox security policy which controls access based on the permissions. As per this Permission based control policy it's mandatory for the user to accept all the permissions for installing an app. This mandatory acceptance of each and every permission by the user for an app made a good platform for the attackers to extract sensitive user mobile information. More than 50% of the apps require the permissions which are irrelevant to their actual functionality.

Peer-review under responsibility of the Organizing Committee of ICRTCSE 2016

doi:10.1016/j.procs.2016.05.150

2. Related Work

Most of the apps from different app stores have their own security policies as a part of security provision to the mobile users. For resource management Apple ios[5], Blackberry[6], Windows Phone[7], Android[8], use Mandatory Access Control(MAC) mechanism. The static permission based model is used by Android by which user acceptance of all requested permission is required. There are several limitations on this permission based model as a result most of the mobile user data is released as data is released or leaked by using permissions we call this as permission leakage. There are various types of Permission leakage attacks [1][9] like collusion attacks[9],Confused deputy attack and Intent spoofing.There are methods and tools available for permission leakage which perform detection and avoidance by static and dynamic analysis.Some of the detection methods are as follows.

TaintDroid[8] is one of the tool which provides analysis of the app by tracking the flow of sensitive information and alerts the user about the behaviour of an app if any of the sensitive user data is leaked by the app. AppFence[9] an application service provider tracks, detects and gives responses without any influence of legitimate users. MockDroid[10] known as modified version of Android Operating systems mocks application utilization to the resource. Mockdroid manily fakes the program data in order not to reveal the user contents which forms a protection. Fire-Droid[11] a policy –based framework utilizes rendering system calls to obligating security policies.

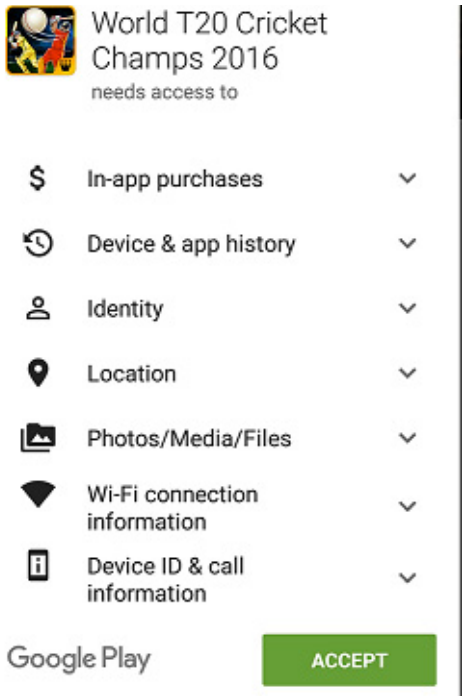
To intensify the basic conventional security policy of Permission control model used in mobile apps, we proposed a framework which gives the user an option to ignore the unnecessary permissions required by an app that would control the sensitive mobile user data leakage. At the beginning we took the datasets from statista portal [4] and Pew Research centre [6]. Atmost there are 1,041,336 apps in pew data set[4] and 235 permissions are observed , 70 allow access to various user information, 165 permissions allow access to device .Then We categorized the apps into 20 different categories taking the apps data, statistics from statista portal[2].Then we calculated the pertinent permissions which require user info for all the category of the apps i.e., for example if app is a games app by categorization ,which are the pertinent permissions from 70 identified user info permissions required to that app are mapped. Based on the datasets for each category of apps we have obliterated the permissions which have less utilization or usage.For all of the 20 categories of apps the required no of permissions for each app and their pertinent permissions based on their utilization or usage are calculated and maintained. To get more sophistication we used RecDroid[11] for the recommendation from experts on permission granting system. By using RecDroid with the final recommended permissions we run the app in probation mode to reduce the risks if there are any false decisions.Inorder to reduce the main permission leakage attack of collusion attack,we have used android apk decompiler[14] and extracted the Android manifest XML file.By using the Android XML manifest file we compared the intent filter components,SharedUid,risk level.Based on some rules we developed by rule based classification to check the

activities in intents we detected permission leakage detection on the apps for example if the User ID in the intent filter is shared for a particular app then that is notified as permission leakage.

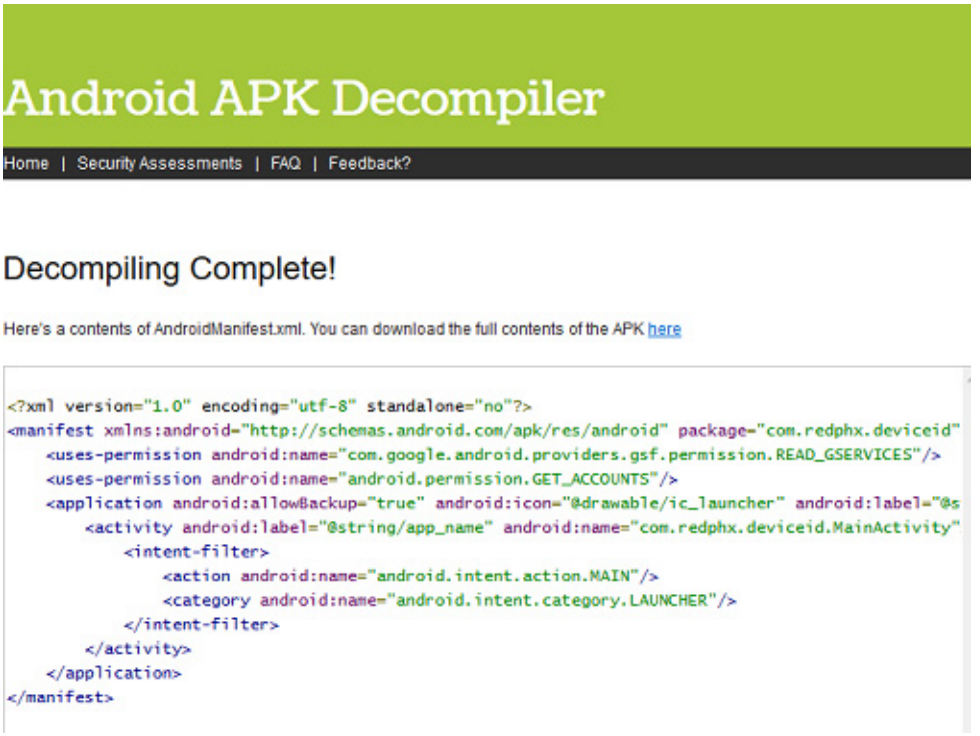
Finally the proposed framework in steps consists of (i)At first taking an app and identifying the pertinent category from 20 categories.(ii) Based on the category of an app the pertinent permissions for app are pointed. (iii) By using RecDroid[13] tool recommendations from experts are taken for separating of permissions and pointed (iv)In the pointed permissions the leakage of permissions is verified by the decompiling app and comparisons of XML file based using rule based classification are done to check any permission leakage activities are there or not.(v)Finally the permissions which are irrelevant to their functionality are notified to the user.

3. Experimental Results

We applied the proposed framework to an app taking from a playstore ,as per the mentioned first step the pertinent category identification from the 20 categories taken from statista portal[4] is done . The 20 categories taken from statista portal are Music , Life Style, News, Books, Games, Education, Business, Food and Drink, Entertainment, Travel, Utilities, Health and Fitness, Sports, Productivity, Finance, Photo and Video, Medical, Social networking, Reference , Navigation



The considered app is World T20 Cricket Champs 2016 , the app comes under sports category. Based on the identified category the pertinent permissions for sports category doesn't necessarily required two permissions In-app permission, Location .By taking the app to Recdroid[11] tool by the expert recommendations one more permission Device ID & Call information is not required.



As per the step four , taking the XML file and using XML comparision tool the intent- filter were detail studied. The permission leakage may happen by the Photos/Medias/Files permission if the same developer app was installed in the device. At this case to avoid such permission leakages the user secured files of photos can be secured by using MockDroid[10] by sending the fake information which could avoid permission leakage attacks like collusion attacks . Finally we identified four permissions were irrelevant to the considered app location, In-app purchases and Device ID and caller information and the sharing of Photos/Medias/Files .

Our Work suggests the user to ignore these irrelevant permissions and install the app using RecDroid[11] in the probation mode .As most of the apps require the storage access the secure photos and files can be shared by faking the information which could avoid permission leakage attacks.

Thus PLANT takes the permissions required by an app and filters the permissions and recommends the user to ignore the irrelevant permissions after the installation which avoids permission leakage that can be done by using the denying option given in permission control settings in the mobile

4. Conclusion

The immense expansion of apps arising several concerns on the security issues of Smartphone users and their privacy.The conventional security policy of permission based control has several limitations by which is giving a platform for the attackers to steal the sensitive and confidential information of the users. To potentiate the

permission based control we developed a framework called PLANT i.e., Permission Leakage AvoidaNce Through Filteration which gives the option to the users to ignore the permissions which are not pertinent to their functionality , by that can stop the leakage of mobile users data.

References

1. Dragos Sbirlea , Michael G. Burke,Salvatore Guarnieri"Automatic Detection of Inter –application Permision Leaks inAndroid applications,Technical Report TR13-02,Dept of CSE, Rice university, January 2013
2. Google: 100000 Android Applications in Market. http: //twitter.com/#!/AndroidDev/status/28701488389
3. Security Report of Top 50 E-commerce Mobile Apps by Appvigil,India August 2015 https://appvigil.co/en/resource/security-report-of-top-e-commerce-mobile-apps/?utm_source=rightbar&utm_medium=blog-widgets&utm_campaign=blog-redirects
4. http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
5. http://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions/#.xwgpjdp:RPIH
6. Pew Research center : http://www.pewinternet.org/interactives/apps-permissions/
7. Claudio Marforio,Francillion Aurelien and Srdjan Capkun "Application Colllusion attack on the permission based security model and its implications for modern smartphone systems.Technical Report 724,Eth zurich, April 2011
8. William Enck, Peter Gilbert, Byung-gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones In Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), October 2010 in Vancouver
9. AppFence : http://www.appfence.sg/
10. Alastair R. Beresford, Andrew Rice ,Nicholas skehin , "MockDroid :trading application functionality on smartphones " Proceeding Hotmobile 11' in Proceeding of 12th Workshop on Mobile Computing systems and Applications Pages 49-54 ACM Newyork,NY,USA ©2011 ISBN: 978-1-4503-0649-2
11. Bahman Rashidi, Carol Fung, RecDroid A Resource Access Permission Control Portal and Recommendation Service for Smartphone Users-ACM SPME'14, September 11, 2014, Maui, Hawaii, USA 978-1-4503-3075-6/14/09, http://dl.acm.org/citation.cfm?doid=2646584.2646586
12. Apple IOS : http://www.apple.com/in/ios/
13. Blackberry :http://global.blackberry.com/en/home.html
14. Windows phone : https://www.microsoft.com/en-us/windows/phones
15. Android : https://www.android.com/